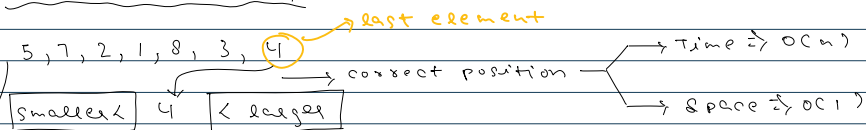


Partition in array

1, 2, 3, 4, 5, 7, 8

→  $n \cdot \log n$  X

→  $n^2$  X

Do not sort the whole array

can be a possibility

3 1 2 4 7 8 5

idx = 0 1 2 3

0 1 2 3 4 5 6

5 7 2 1 8 3 4

si = 0

ei = 6

Pivot = arr[ei]

idx = si

2 7 5 1 8 3 4

2 1 5 7 8 3 4

2 1 3 7 8 5 4

swap  $\Rightarrow$  arr[i], arr[idx]

↓

arr[i] < pivot

idx++

2 1 3 7 8 5 4

↓

idx

arr[idx], pivot  $\Rightarrow$  swap

2 1 3 4 8 5 7

0 1 2 3 4 5 6

5 7 2 1 8 3 4

si = 0

ei = 6

Pivot = arr[ei]

idx = si

idx = 0 1 2 3

2 7 5 1 8 3 4

2 1 5 7 8 3 4

2 1 3 7 8 5 4

arr[i] < pivot

swap arr[idx], arr[i]

idx++

2 1 3 7 8 5 4

↓

idx

arr[idx], pivot  $\Rightarrow$  swap

2 1 3 4 8 5 7

```
public class Main {
    public static void main(String[] args) {
        int[] arr = {5, 7, 2, 1, 8, 3, 4};
        display(arr);
        int idx = partition(arr, 0, arr.length-1);
        System.out.println("partition index: "+idx);
        display(arr);
    }
}
```

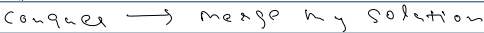
```
public static int partition(int[] arr, int si, int ei){
    int pivot = arr[ei];
    int idx = si;
    for(int i=si; i<ei; i++){
        if(arr[i] < pivot){
            // swap  $\Rightarrow$  arr[i], arr[idx]
            int temp = arr[i];
            arr[i] = arr[idx];
            arr[idx] = temp;

            idx++;
        }
        display(arr);
    }
}
```

}

}

## Divide and Conquer



## Merge Sort



Time complexity  $\propto$  height of the tree

$$\frac{10}{10} = 1$$

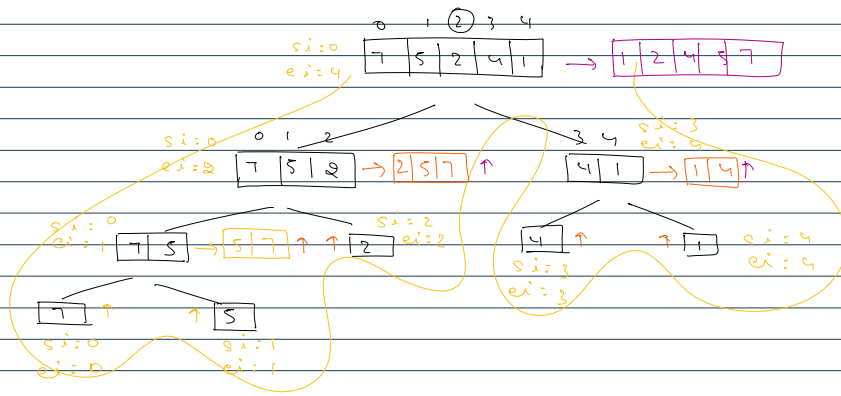
$$n = 2^k$$

merge 2 sorted  
array

$$K = \log_2 n$$

$$\begin{array}{r} 11 \\ 604 \end{array}$$

$\Omega_{\text{pole}} = 1, 0(\omega)$



```

public class Main {
    public static void main(String[] args) {
        int[] arr = {5, 7, 2, 1, 8, 3, 4};
        display(arr);
        int[] sortedArray = sort(arr, 0, arr.length-1);
        display(sortedArray);
    }

    public static int[] sort(int[] arr, int si, int ei){
        if(si == ei){
            int[] bs = { arr[si] };
            return bs;
        }
        int mid = (si + ei)/2;
        int[] first = sort(arr, si, mid); //1st sorted array
        int[] second = sort(arr, mid+1, ei); //2nd sorted array
        return merge(first, second);
    }

    public static void display(int[] arr){
        for(int i=0; i<arr.length; i++){
            System.out.print(arr[i] + " ");
        }
        System.out.println();
    }

    public static int[] merge(int[] arr1, int[] arr2) {
        int n = arr1.length;
        int m = arr2.length;

        int[] arr = new int[n+m];

        int i = 0;
        int j = 0;
        int k = 0;

        while(i<n && j<m){
            if(arr1[i] < arr2[j]){
                arr[k] = arr1[i];
                i++;
            }
            else{
                arr[k] = arr2[j];
                j++;
            }
            k++;
        }

        while(i<n){
            arr[k] = arr1[i];
            k++;
            i++;
        }
        while(j<m){
            arr[k] = arr2[j];
            k++;
        }
    }
}

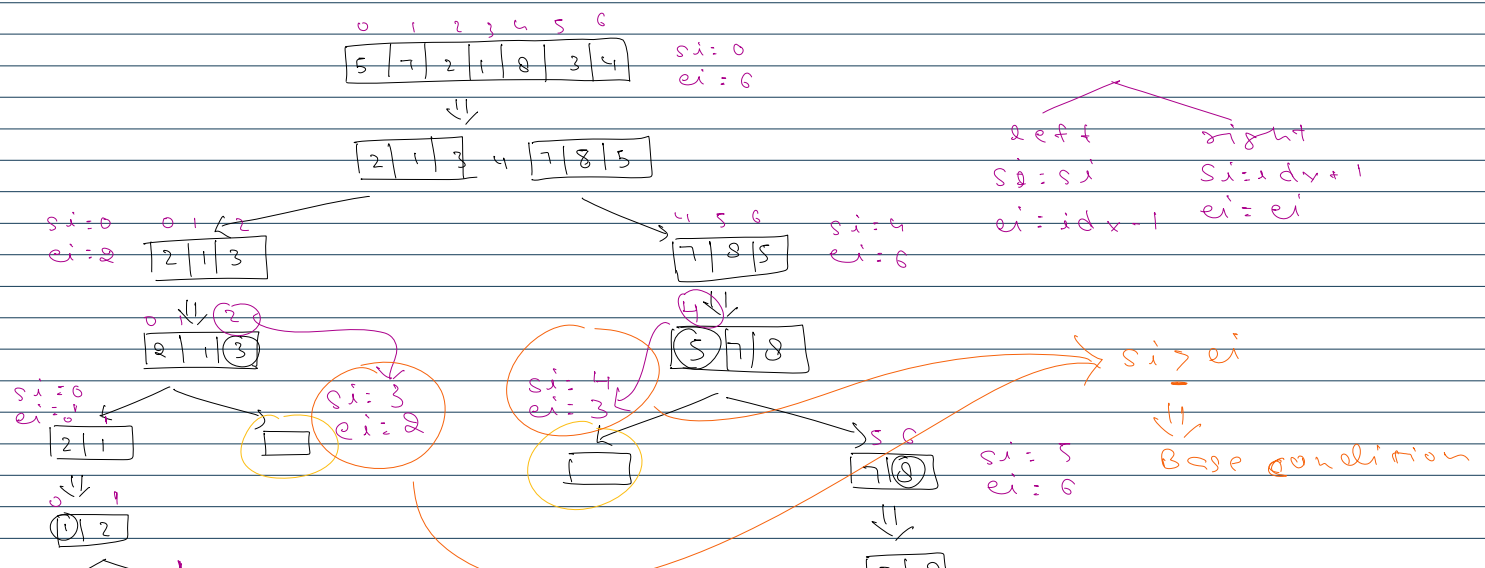
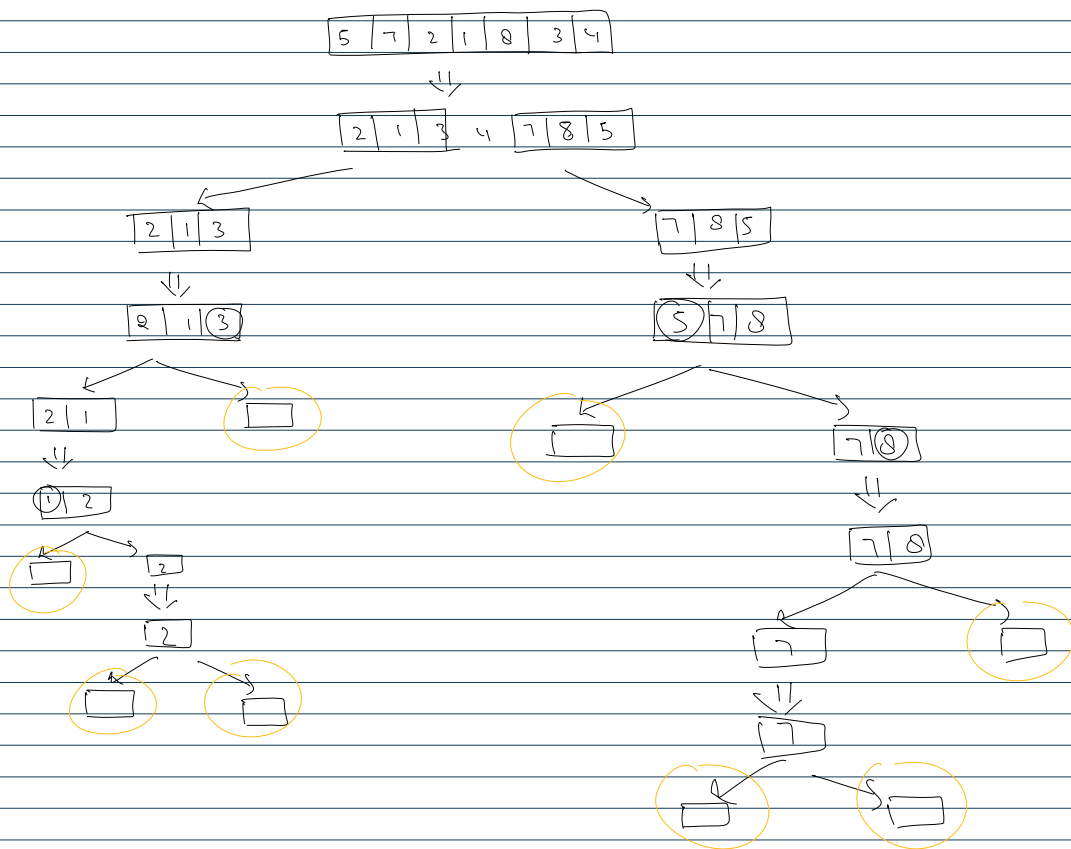
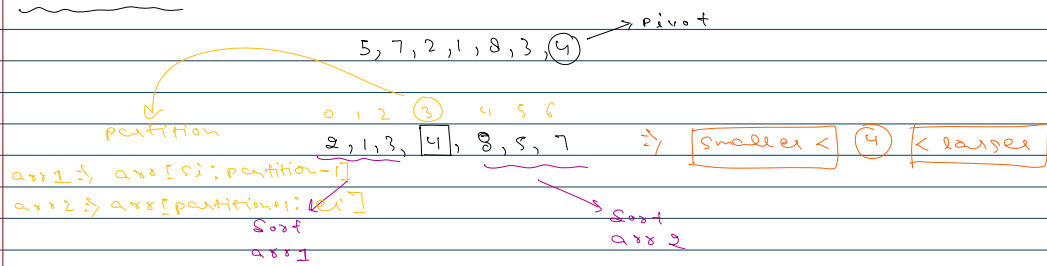
```

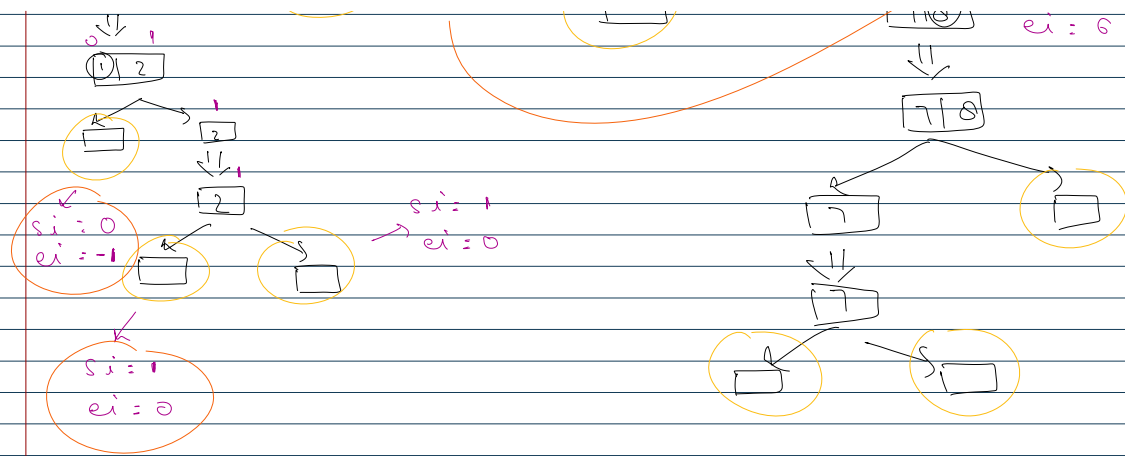
```

j++;
}
return arr;
}
}

```

## Quick Sort



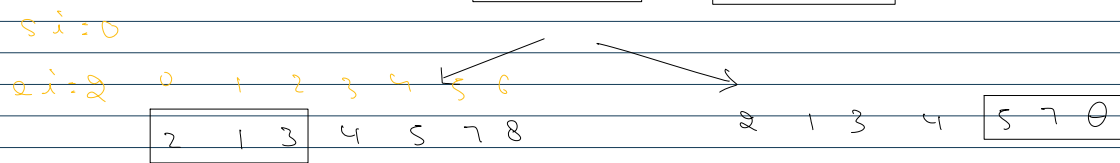


0	1	2	3	4	5	6
5	7	2	1	8	3	(4)

0	1	2	(3)	4	5	6
2	1	3	(4)	5	7	8

$si=0$   
 $ei=6$



```

public class Main {
    public static void main(String[] args) {
        int[] arr = {5, 7, 2, 1, 8, 3, 4};
        display(arr);
        sort(arr, 0, arr.length-1);
        display(arr);
    }

    public static void sort(int[] arr, int si, int ei){
        if(si >= ei){
            return;
        }
        int idx = partition(arr, si, ei);
        sort(arr, si, idx-1);
        sort(arr, idx+1, ei);
    }

    public static void display(int[] arr){
        for(int i=0; i<arr.length; i++){
            System.out.print(arr[i] + " ");
        }
    }
}

```

```

    }
    System.out.println();
}

public static int partition(int[] arr, int si, int ei){
    int pivot = arr[ei];
    int idx = si;
    for(int i=si; i<ei; i++){
        if(arr[i] < pivot){
            // swap => arr[i], arr[idx]
            int temp = arr[i];
            arr[i] = arr[idx];
            arr[idx] = temp;

            idx++;

            display(arr);
        }
    }

    // swap => arr[idx], pivot(arr[ei])
    int temp = arr[ei];
    arr[ei] = arr[idx];
    arr[idx] = pivot;
    return idx;
}

```

	merge	quick
	sort	sort
Worst	$n \log n$	$n^2$
Best / average	$n \log n$	$n^2$
Space	O(1)	O(1)
Inplace	X	✓
	Divide	Divide

} → Randomized quick sort

```

public class Main {
    public static void main(String[] args) {

        int low = 10;
        int high = 100;

        Random rn = new Random();
        for(int i=0; i<20; i++){
            int x = rn.nextInt(high-low+1) + low; //91
            System.out.println(x);
        }

    }
}

```